nnygk5okg

August 21, 2024

1 SVM Practive - Answer

Aug 21, 2024

1.1 1. Bài toán

Phân loại văn bản sử dụng Naive Bayes

Mục tiêu:

- Xây dựng được mô hình SVM sử dụng thư viện sklearn.
- Ứng dụng và hiểu cách áp dụng mô hình SVM vào giải quyết bài toán thực tế (*ví dụ: phân loại văn bản*).
- Sử dụng độ đo Accuracy để đánh giá chất lượng mô hình.

Vấn đề: * Có một tập các văn bản dạng text không có nhãn, làm sao để biết văn bản này thuộc về thể loại nào, pháp luật, đời sống, văn học, thể thao,...

Dữ liệu: * Tập các văn bản và nhãn tương ứng của từng văn bản trong một khoảng thời gian. * Tập các nhãn - 10 nhãn văn bản:

> Giải trí, Khoa học - Công nghệ, Kinh tế, Pháp luật, Sức khỏe, Thể thao, Thời sự, Tin khác, Độc giả, Đời sống - Xã hội.

Ví dụ văn bản nhãn thể thao:

> "Dân_trí Real Madrid đã dẫn trước trong cả trận đấu , nhưng họ vẫn phải chấp_nhận bị Dortmund cầm hòa 2-2 ở Bernabeu . Real Madrid chấp_nhận đứng thứ_hai ở bảng F Champions League …"

Bài toán: Phân loại

- Input: n vector mã hóa của các văn bản - ma trận $X = [x_1, x_2, ...x_n]$
- Output: nhãn y là 1 trong 10 nhãn trên

1.2 2. Import các thư viện cần thiết, cài thêm một số thư viện chưa sẵn có

[ ]:
```
# Cài đặt thư viện xử lý ngôn ngữ cho tiếng Việt!
!pip install pyvi
```

[ ]:
```
import matplotlib.pyplot as plt
import numpy as np
from sklearn.model_selection import learning_curve

from sklearn.datasets import load_files
from pyvi import ViTokenizer  # thư viện tách từ Tiếng Việt

from sklearn import svm
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer, TfidfTransformer
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.pipeline import Pipeline
from sklearn.metrics import accuracy_score

%matplotlib inline
```

1.3 3. Load dữ liệu từ thư mục đã crawl từ trước

Cấu trúc thư mục như sau: - data/news_1135/ - Kinh tế/ - bài báo 1.txt - bài báo 2.txt - Pháp luật/ - bài báo 3.txt - bài báo 4.txt

[ ]:
```
%cd /content/drive/MyDrive/Code_VinBigData_2024
```

[ ]:
```
data_train = load_files(container_path="data/data/news_1135/", encoding="utf-8")
### bài tập ###
# yêu cầu: In ra các kết quả sau:
#     - Tên 10 file dữ liệu đầu.
#     - Tổng số file dữ liệu.
#     - Danh sách nhãn và id tương ứng.
# gợi ý: tự làm
################
# code

################
```

1.4 4. Tiền xử lý dữ liệu đưa dữ liệu từ dạng text về dạng ma trận

[ ]:
```
### bài tập ###
# yêu cầu: Tiền xử lý dữ liệu text về dạng ma trận, sử dụng TF-IDF.
# gợi ý: tương tự bài thực hành về NaiveBayes.
################
# code
# load dữ liệu các stopwords

################
print("10 từ đầu tiên trong từ điển:")
i = 0
```

```
for k, v in module_count_vector.vocabulary_.items():
    i += 1
    print(i, ": ", (k, v))
    if i > 9:
        break
```

[ ]:
```
### Example
corpus = [
    "This is the first document.",
    "This document is the second document.",
    "And this is the third one.",
    "Is this the first document?",
]
vectorizer = CountVectorizer()
X = vectorizer.fit_transform(corpus)
# vectorizer.get_feature_names_out()

for k, v in vectorizer.vocabulary_.items():
    i += 1
    print(i, ": ", (k, v))
    # if i > 10:
    #     break


print(X.toarray())

vectorizer2 = CountVectorizer(analyzer="word", ngram_range=(2, 2))
X2 = vectorizer2.fit_transform(corpus)
# vectorizer2.get_feature_names_out()
```

[ ]:
```
### bài tập ###
# yêu cầu: Tiền xử lý với TfidfVectorizer, in ra 10 từ đầu tiên trong từ điển.
# gợi ý: https://scikit-learn.org/stable/modules/generated/sklearn.
 ↪feature_extraction.text.TfidfVectorizer.html
###############
# code
###############
```

[ ]:
```
### bài tập ###
# yêu cầu: Tìm ra top 10 từ có giá trị TF-IDF cao nhất.
# gợi ý: tự làm
###############
# code
###############
```

1.5 5. Chia dữ liệu làm 2 phần training và testing

- Training chiếm 80 % dữ liệu
- Testing chiếm 20 % dữ liệu

[ ]:
```
# from sklearn.model_selection import ShuffleSplit

test_size = 0.2
# cv = ShuffleSplit(n_splits=10, test_size=0.2, random_state=0)
X_train, X_test, y_train, y_test = train_test_split(data_preprocessed,␣
 ↪data_train.target, test_size=test_size, random_state=0)

print("Dữ liệu training = ", X_train.shape, y_train.shape)
print("Dữ liệu testing = ", X_test.shape, y_test.shape)
```

1.6 6. Training SVM model

Sử dụng thư viện sklearn để xây dựng mô hình:

`svm.SVC(kernel='linear', C=1.0)`: chọn hàm nhân phân tách là linear, tham số C=1.0

[ ]:
```
print("- Training ...")
print("- Train size = {}".format(X_train.shape))
model = svm.SVC(kernel="linear", C=1.0)
model.fit(X_train, y_train)
print("- model - train complete")
```

1.7 7. Testing Naive Bayes model

- Thực hiện dự đoán nhãn cho từng văn bản trong tập test
- Độ đo đánh giá: > accuracy = tổng số văn bản dự đoán đúng / tổng số văn bản có trong tập test

[ ]:
```
print("- Testing ...")
y_pred = model.predict(X_test)
print("- Acc = {}".format(accuracy_score(y_test, y_pred)))
```

[ ]:
```
### bài tập ###
# yêu cầu: Thực hiện lại các bước huấn luận, kiểm thử mô hình với hàm phân tách␣
 ↪'rbf' (radial basis function), hay hàm Gaussian.
# gợi ý: tự làm
################
# code
################
```

[ ]:
```
### bài tập ###
# yêu cầu: Dự đoán nhãn cho văn bản: "Mệt mỏi vì hóa đơn tiền điện tăng cao?
```

```
#                                       Muốn tận hưởng không gian mát lạnh mà␣
 ↪không lo tốn kém?
#                                       Video này sẽ bật mí cho bạn những bí␣
 ↪quyết sử dụng điều hòa Inverter hiệu quả nhất.
#                                       Từ cách chọn chế độ, cài đặt nhiệt độ đến␣
 ↪bảo dưỡng máy,
#                                       tất cả sẽ được giải đáp chi tiết.
#                                       Đừng bỏ lỡ cơ hội tiết kiệm hàng triệu␣
 ↪đồng mỗi năm!".
#          So sánh kết quả dự đoán của 2 mô hình đã huấn luyện với 2 hàm phân␣
 ↪tách "linear" và "rbf"
# gợi ý: tự làm
###############
# code

###############
```

1.8 8. Bài tập bổ sung:

8.1 Thử nghiệm các tham số

- Các tham số với giá trị khác nhau có thể ảnh hưởng để kết quả học
- Cần thử nghiệm kỹ lượng để đưa ra kết quả khách quan: tham số C, gamma, kernel.
  - Chọn mô hình với bộ tham số cho kết quả tốt nhất
- Gợi ý:
  - https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html
  - Sử dụng grid search

Bài tập: Vẽ Learning curve khảo sát Acc của SVM-linear với tham số C thay đổi

[ ]:
```
list_C = [0.001, 0.01, 0.1, 1, 5.0, 10.0, 100]
list_acc = []
title = "Learning Curves SVM, Linear kernel, change C"

# duyệt qua mảng các giá trị của tham số C
for i, C in enumerate(list_C):
    # Với từng giá trị C nhận được,
    # thực hiện build model và training cross-validate
    # vẽ kết quả tìm được lên đồ thị đường.
    model = svm.SVC(kernel="linear", C=C)
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    list_acc.append(accuracy_score(y_test, y_pred))
```

[ ]:
```
import seaborn as sns

fig = sns.lineplot(x=list(range(0, 7)), y=list_acc)
fig.set_xticks(range(0, 7))
```

```
fig.set_xticklabels([0.001, 0.01, 0.1, 1, 5.0, 10.0, 100])
```

[ ]:
```
# hàm sinh id màu
def get_cmap(n):
    return "C" + str(n)


# Hàm thực hiện training model, crossvalidate và vẽ lên đồ thị sử dụng mat␣
 ↪libplot
def plot_learning_curve(estimator, title, label_curve, X, y, ylim=None,␣
 ↪cv=None, n_jobs=1, train_sizes=np.linspace(0.1, 1.0, 5), new_plot=False,␣
 ↪idx_color=0):
    # Khởi tạo bức ảnh mới với thư viện plot lib
    if new_plot:
        # plt.figure()
        plt.title(title)
        plt.xlabel("Training examples")
        plt.ylabel("Accuracy")
        plt.grid()

    # chú thích nếu có
    if ylim is not None:
        plt.ylim(*ylim)

    # thực hiện training model, ghi nhận các giá trị trong quá trình training
    # cv = số fold cross validate, số phần bộ dữ liệu được chia để thực hiện␣
 ↪training testing.
    # train_sizes = mảng tỉ lệ, các tỉ lệ được hệ thống chọn làm điểm dừng để␣
 ↪thực hiện 1 testing
    #  train_sizes = [0.3, 0.5] => hệ thống lấy 30 % dữ liệu để train và thực␣
 ↪hiện test, tương tự 50 % ..
    # scoring = hàm mục tiêu để đánh giá chất lượng mô hình và vẽ lên đồ thị
    train_sizes, train_scores, test_scores = learning_curve(estimator, X, y,␣
 ↪cv=cv, n_jobs=n_jobs, train_sizes=train_sizes, scoring="accuracy")

    # Lấy trung bình cộng các giá trị output của các fold
    train_scores_mean = np.mean(train_scores, axis=1)
    train_scores_std = np.std(train_scores, axis=1)
    test_scores_mean = np.mean(test_scores, axis=1)
    test_scores_std = np.std(test_scores, axis=1)

    # random 1 màu để vẽ
    color = get_cmap(idx_color)

    # thực hiện vẽ các giá trị số lên đồ thị với màu vừa được random
```

```
    plt.fill_between(train_sizes, test_scores_mean - test_scores_std,␣
 ↪test_scores_mean + test_scores_std, alpha=0.1, color=color)
    plt.plot(train_sizes, test_scores_mean, "o-", color=color,␣
 ↪label=label_curve)

    plt.legend(loc="best")
    return plt
```

[ ]:
```
list_C = [0.001, 0.01, 0.1, 1, 5.0, 10.0, 100]
# model title
title = "Learning Curves SVM, Linear kernel, change C"

# duyệt qua mảng các giá trị của tham số C
for i, C in enumerate(list_C):
    # Với từng giá trị C nhận được,
    # thực hiện build model và training cross-validate
    # vẽ kết quả tìm được lên đồ thị đường.
    text_clf = Pipeline(
        [
            ("clf", svm.SVC(kernel="linear", C=C)),  # mô hình svm với tham số C
        ]
    )

    plt = plot_learning_curve(text_clf, title, "C = %.3f" % (C),␣
 ↪data_preprocessed, data_train.target, (0.0, 1.01), cv=10, n_jobs=-1,␣
 ↪idx_color=i, new_plot=i == 0)

# lưu hình ảnh ra file
# plt.savefig('images/changeC.png', bbox_inches='tight')
plt.show()
```

[ ]:
```
### bài tập ###
# yêu cầu: Đánh giá sự thay đổi của hàm phân tách ảnh hưởng thế nào đến độ␣
 ↪chính xác của mô hình. Vẽ biểu đồ minh họa.
# gợi ý: Xét các hàm phân tách được định nghĩa trước trong svm.SVC() ('linear',␣
 ↪'poly', 'rbf')
#        Tương tự thay đổi về giá trị C
###############
# code

###############
```

Tunning model: Sử dụng GridSearchCV để tìm bộ tham số tốt nhất

[ ]:
```
params_grid = {"C": [0.001, 0.01, 0.1, 1, 10, 100], "gamma": [0.0001, 0.001, 0.
 ↪01, 0.1], "kernel": ["linear", "rbf", "poly"]}
```

```
model = svm.SVC()
# Create the GridSearchCV object
best_model = GridSearchCV(model, params_grid, cv=4, n_jobs=-1,␣
 ↪scoring="accuracy")

# Fit the data with the best possible parameters
best_model.fit(X_train, y_train)

# Print the best estimator with it's parameters
print(best_model.best_params_)
print(best_model.best_estimator_)
# Test best_model
print("Testing")
y_pred = best_model.predict(X_test)
print(accuracy_score(y_test, y_pred))
```